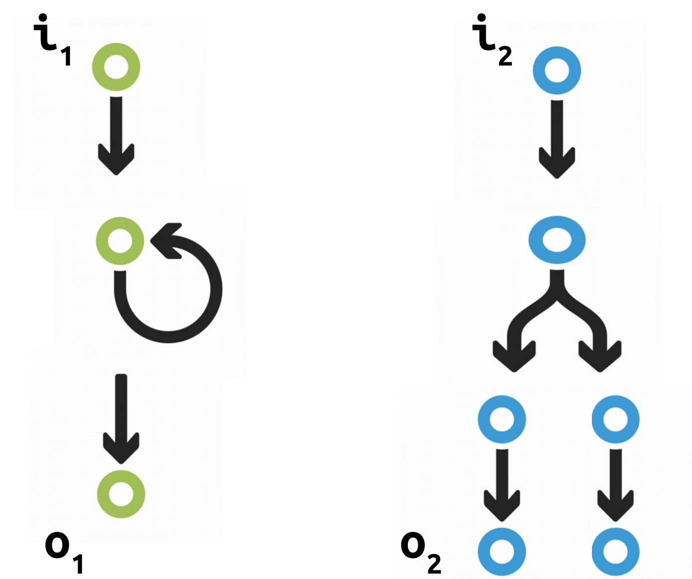
Enhancing Predicate Pairing with Abstraction for Relational Verification

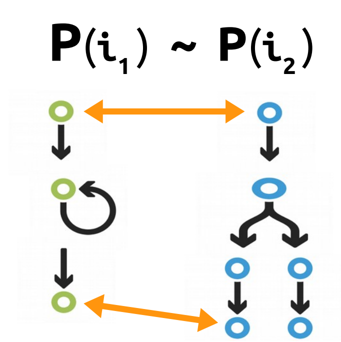
E. De Angelis[1], F. Fioravanti[1],
A. Pettorossi[2], and M. Proietti[3]

[1] University of Chieti-Pescara `G. d'Annunzio'
[2] University of Rome `Tor Vergata'
[3] CNR - Istituto di Analisi dei Sistemi ed Informatica

Relational verification (1)

two different program executions





Relational property

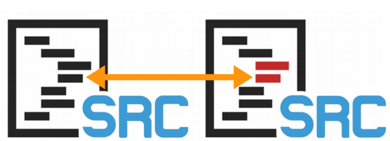
Program Monotonicity

If **P** terminates on the input i_1 producing o_1 &
P terminates on the input i_2 producing o_2 &
i_1 is less than i_2
then
o_1 is less than o_2

Relational verification (2)

two different programs

$P_1(i_1) \sim P_2(i_2)$



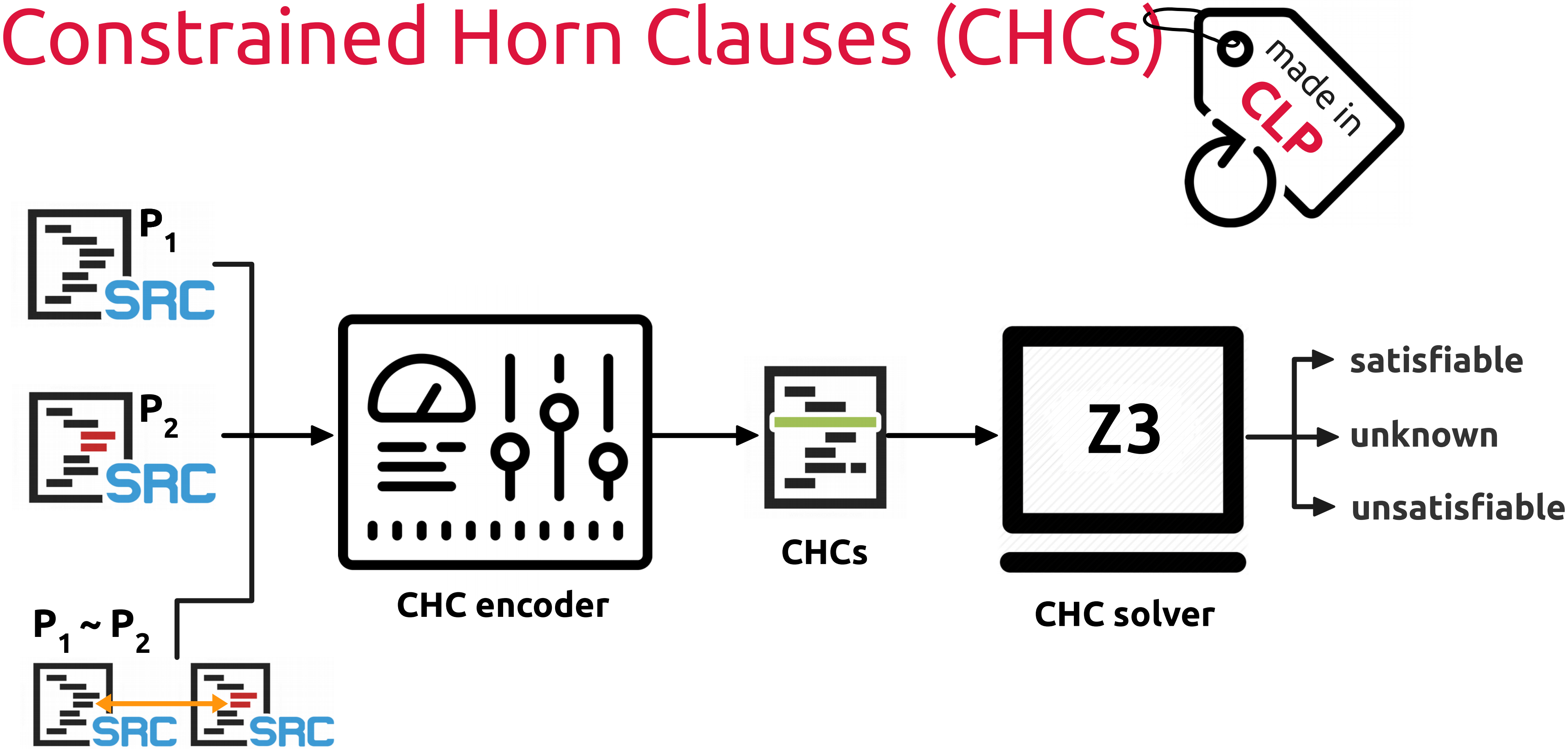
Relational property

Program Equivalence

If P_1 terminates on the input i_1 producing o_1 &
P_2 terminates on the input i_2 producing o_2 &
i_1 equals to i_2
then
o_1 equals to o_2

Relational verification

Constrained Horn Clauses (CHCs)



The relational property holds

if & only if

{ CHCs for $\mathbf{P_1 \sim P_2}$ } U { CHCs for $\mathbf{P_1}$ } U { CHCs for $\mathbf{P_2}$ } is **satisfiable**

Example

CHC encoding

P1

```
int a, b, x, y;
```

```
while (a < b) {
    x = x+a;
    y = y+x;
    a = a+1;
}
```

Example
CHC encoding

P1

`int a, b, x, y;`	A1, B1, X1, and Y1	**input** values
	A1’, B1’, X1’, and Y1’	**output** values

```
while (a < b) {
   x = x+a;
   y = y+x;
   a = a+1;
}
```

Example

CHC encoding

P1

`int a, b, x, y;`	A1, B1, X1, and Y1	**input** values
	A1’, B1’, X1’, and Y1’	**output** values
`while (a < b) { x = x+a; y = y+x; a = a+1; }`	**P1whl**(A1,B1,X1,Y1, A1’,B1’,X1’,Y1’)	**input/output relation**

Example

CHC encoding

P1

`int a, b, x, y;`	A1, B1, X1, and Y1	**input** values
	A1', B1', X1', and Y1'	**output** values

`while (a < b) {`
`  x = x+a;`
`  y = y+x;`
`  a = a+1;`
`}`
	P1whl(A1,B1,X1,Y1, A1',B1',X1',Y1') ← A1≤B1-1, X1''=A1+X1, Y1''=Y1+X1, A1''=A1+1, **P1whl**(A1'',B1,X1'',Y1'', A1',B1',X1',Y1')
	
	P1whl(A1,B1,X1,Y1, A1,B1,X1,Y1) ← A1≥B1
		input/output relation

Example

CHC encoding

P1

```
while (a < b) {
   x = x+a;
   y = y+x;
   a = a+1;
}
```

P2

```
if (a < b) {
   x = x+a;
   while (a < b-1) {
      y = y+x;
      a = a+1;
      x = x+a;
   }
   y = y+x;
   a = a+1;
}
```

$$\mathbf{P1whl}(A1,B1,X1,Y1,A1',B1',X1',Y1') \leftarrow$$
$$A1 \leq B1-1,\ X1''=A1+X1,\ Y1''=Y1+X1,\ A1''=A1+1,$$
$$\mathbf{P1whl}(A1'',B1,X1'',Y1'',A1',B1',X1',Y1')$$

$$\mathbf{P1whl}(A1,B1,X1,Y1,A1,B1,X1,Y1) \leftarrow A1 \geq B1$$

Example

CHC encoding

P1

```
while (a < b) {
   x = x+a;
   y = y+x;
   a = a+1;
}
```

P1whl(A1,B1,X1,Y1,A1’,B1’,X1’,Y1’) ←
 A1≤B1-1, X1”=A1+X1, Y1”=Y1+X1, A1”=A1+1,
 P1whl(A1”,B1,X1”,Y1”,A1’,B1’,X1’,Y1’)

P1whl(A1,B1,X1,Y1,A1,B1,X1,Y1) ← A1≥B1

P2

```
if (a < b) {
   x = x+a;
   while (a < b-1) {
      y = y+x;
      a = a+1;
      x = x+a;
   }
   y = y+x;
   a = a+1;
}
```

P2ite(A2,B2,X2,Y2,A2’,B2’,X2’,Y2’) ←
 A2≤B2-1, X2’’=X2+A,
 P2whl(A2,B2,X2’’,Y2,A2’,B2’,X2’,Y2’)

P2ite(A2,B2,X2,Y2,A2,B2,X2,Y2) ← A2≥B2

P2whl(A2,B2,X2,Y2,A2’,B2’,X2’,Y2’) ←
 A2≤B2-2, Y2’’=Y2+X2, A2’’=A2+1, X2’’=X2+A2,
 P2whl(A2’’,B2,X2’’,Y2’’,A2’,B2’,X2’,Y2’)

P2whl(A2,B2,X2,Y2,A2’,B2,X2,Y2’) ←
 A2≥B2-1, Y2’=Y2+X2, A2’=A2+1

Example
equivalence

P1

P1whl(A1,B1,X1,Y1,A1',B1',X1',Y1') ←
A1≤B1-1, X1''=A1+X1, Y1''=Y1+X1,…,
P1whl(A1'',B1,X1'',Y1'',A1',B1',X1',Y1')
P1whl(A1,B1,X1,Y1,A1,B1,X1,Y1) ←
A1≥B1

?
X1'=X2'

P2

P2ite(A2,B2,X2,Y2,A2',B2',X2',Y2') ←
A2≤B2-1, X2''=X2+A,
P2whl(A2,B2,X2'',Y2,A2',B2',X2',Y2')
P2ite(A2,B2,X2,Y2,A2,B2,X2,Y2) ←
A2≥B2
P2whl(A2,B2,X2,Y2,A2',B2',X2',Y2') ←
A2≤B2-2,Y2''=Y2+X2,A2''=A2+1,…,
P2whl(A2'',B2,X2'',Y2'',A2',B2',X2',Y2')
P2whl(A2,B2,X2,Y2,A2',B2,X2,Y2') ←
A2≥B2-1, Y2'=Y2+X2, A2'=A2+1

A1=A2, B1=B2, X1=X2, Y1=Y2,
P1whl(A1,B1,X1,Y1,A1',B1',X1',Y1'), **P2ite**(A2,B2,X2,Y2,A2',B2',X2',Y2') →
X1'=X2'

Example

equivalence

A1=A2, B1=B2, X1=X2, Y1=Y2,
P1whl(A1,B1,X1,Y1, A1’,B1’,X1’,Y1’), **P2ite**(A2,B2,X2,Y2, A2’,B2’,X2’,Y2’) →
X1’=X2’

false ← A1=A2, B1=B2, X1=X2, Y1=Y2, X1’ ≠ X2’,
P1whl(A1,B1,X1,Y1, A1’,B1’,X1’,Y1’), **P2ite**(A2,B2,X2,Y2, A2’,B2’,X2’,Y2’)

Satisfiability of CHCs

State-of-the-art solvers for CHCs with **Linear Integer Arithmetic** (**LIA**) look for **models** of **single atoms**:

to prove that **P1whl** and **P2ite** are equivalent solvers should discover **quadratic relations**.

$$X_1' = X_1 + \frac{(B_1 - A_1) \cdot (B_1 + A_1 - 1)}{2}$$

Satisfiability of CHCs

State-of-the-art solvers for CHCs with **Linear Integer Arithmetic** (**LIA**) look for **models** of **single atoms**:

to prove that **P1whl** and **P2ite** are equivalent solvers should discover **quadratic relations**.

"**solution**"
buy a smarter solver, that is,
a solver for non-linear integer arithmetic

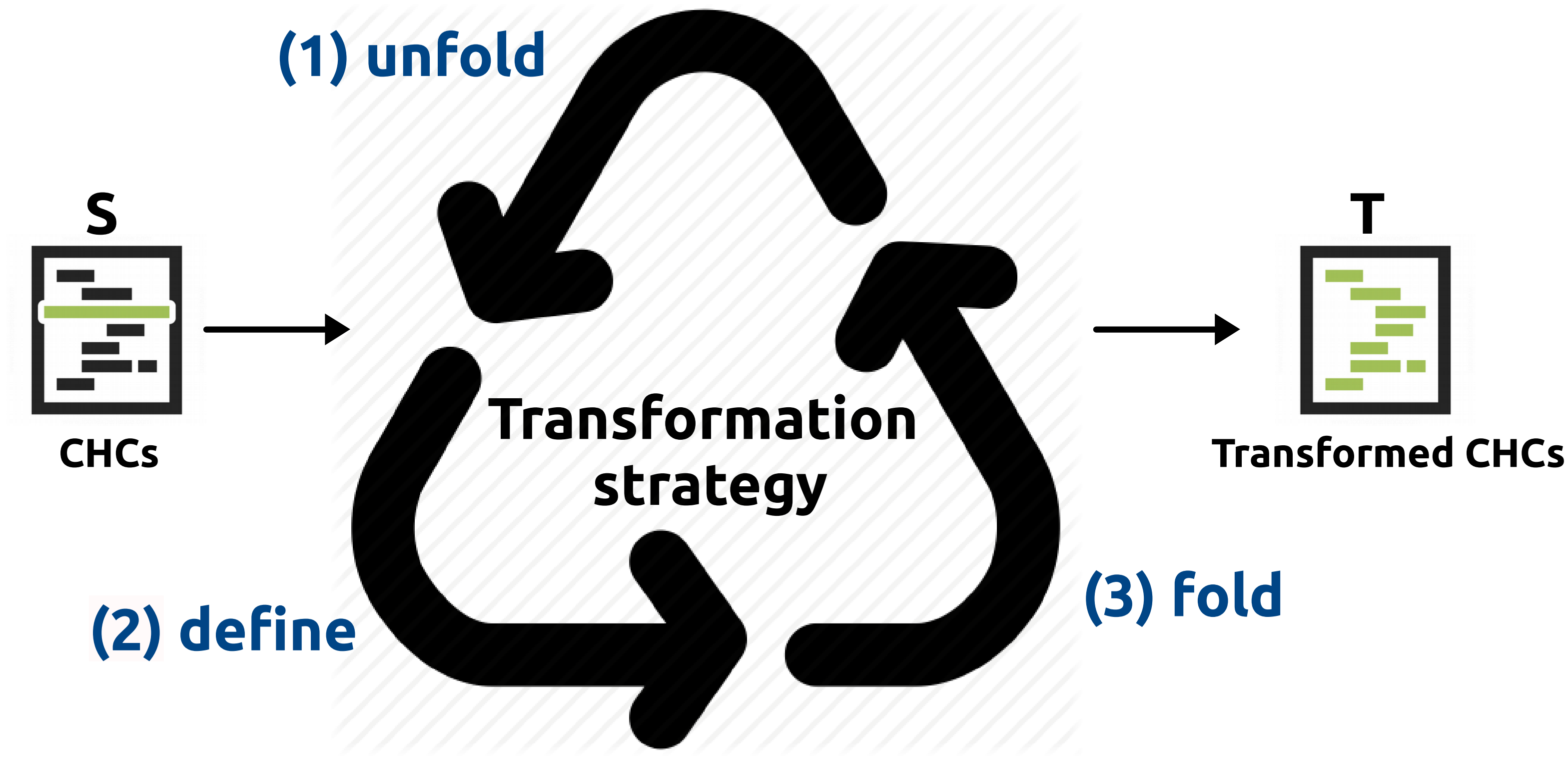
drawback:
satisfiability of constraints is **undecidable**
(decide satisfiability of Diophantine equations)

Our contribution

To make the conjunction of **P1whl** and **P2ite** enables solvers to look for models of their **conjunction**.

To discover **linear relations** among the **arguments** of **P1whl** and **P2ite** may help solvers to prove the satisfiability of CHCs.

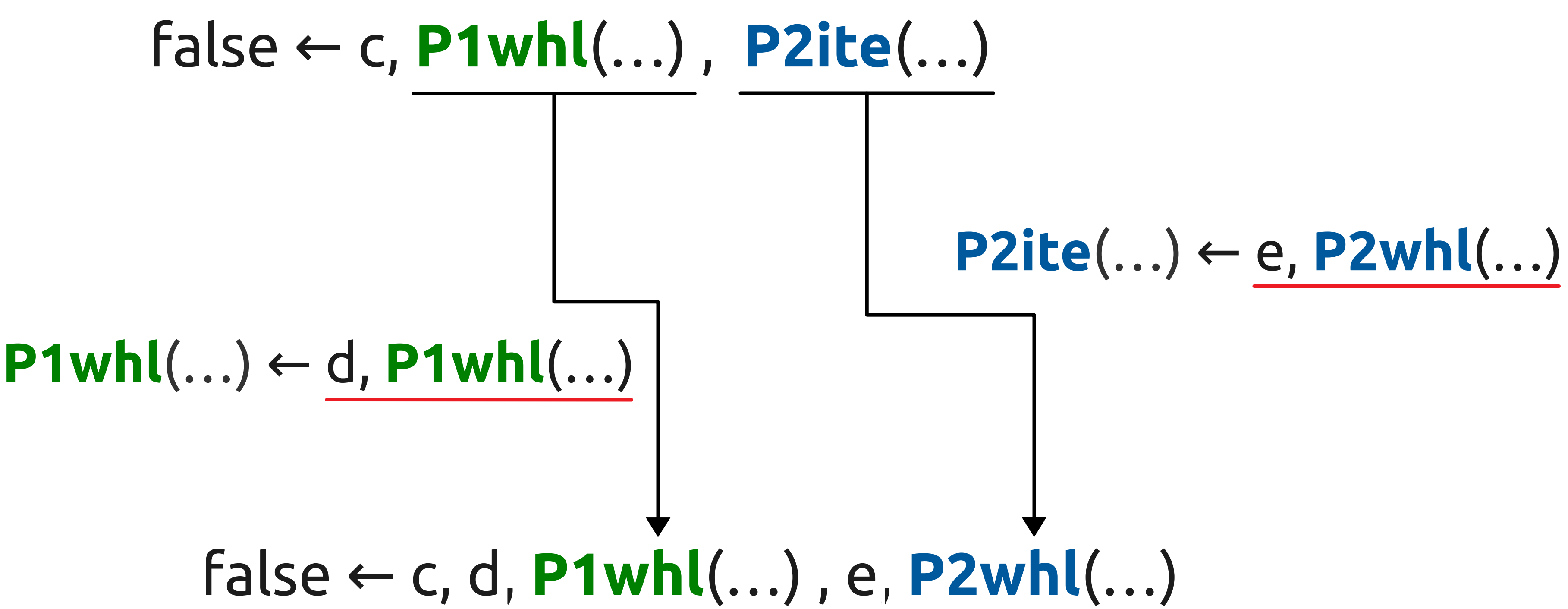
Rule-based transformation of CHCs



S is satisfiable if & only if **T** is satisfiable

Transformation strategy (1)

unfold the atoms **P1whl**(…) and **P2ite**(…), that is, replace **P1whl**(…) and **P2ite**(…) with their bodies



Transformation strategy (2)

Given a clause obtained by unfolding

false ← c, d, **P1whl**(…) , e, **P2whl**(…)

define a new predicate

P1whlP2whl(…) ← **P1whl**(…), **P2whl**(…)

equivalent to the conjunction **P1whl**(…), **P2whl**(…)

Transformation strategy (3)

fold, that is, replace the atoms **P1whl**(…) and **P2ite**(…) with the new predicate **P1whlPwhl**(…)

false ← c, d, **P1whl**(…) , e, **P2whl**(…)

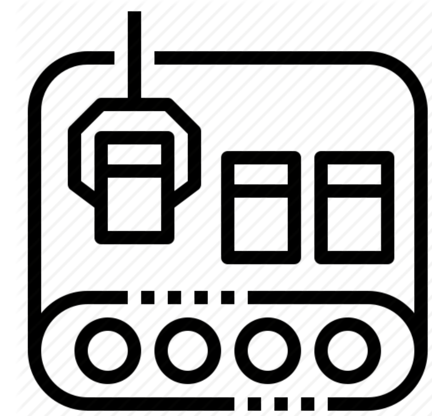
false ← c, d, e , **P1whlPwhl**(…)

Solvers will look for **models of the conjunction**.

Transformation strategy

Assembling new definitions

The transformation strategy is parametric with respect to a **partition operator** that selects the atoms to create new predicate definitions:



one atom → **Specialization**

two atoms → **Predicate Pairing** (**PP**)

Definitions with three or more atoms can be obtained by **iterating** PP.

Enhancing predicate pairing

Abstraction-based **Predicate Pairing** (**APP**)

P1whlP2whl(...) ← **a**, **P1whl**(...), **P2whl**(...)

the definition is augmented with a constraint **a** representing **some** relations among the arguments of **P1whl** and **P2whl**.

The new constraint **a** is an abstraction of the constraint **c**, **d**, **e**

(**c**, **d**, **e**) → **a**

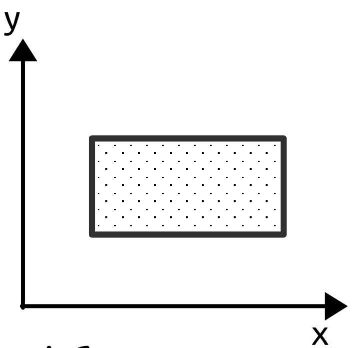
occurring in the clause obtained by unfolding:

false ← c, d, **P1whl**(…) , e, **P2whl**(…)

Enhancing predicate pairing

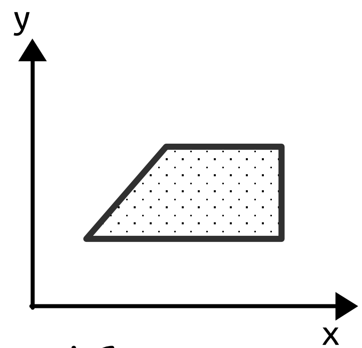
abstract domains

Boxes



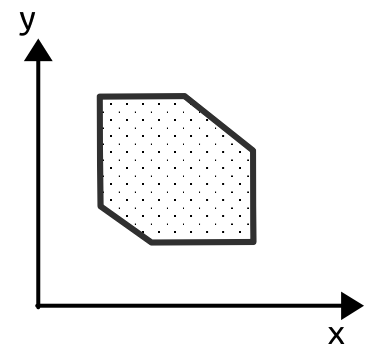
x >= 2
x <= 10
y >= 3
y <= 7

Bounded Difference Shapes



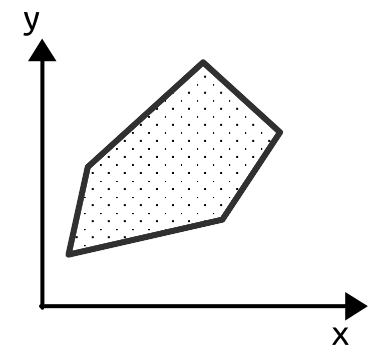
x - y >= 0
y <= 6
y >= 3
x <= 10

Octagonal Shapes



x+y >= 3
x+y <= 12
x >= 2
x <= 10
y <= 9
y >= 3

Convex Polyhedra



x+y <= 35
y >= x - 5
7x <= y+95
3y >= x+7
y <= 4x

The **transformation strategy** is parametric with respect to the abstract constraint domain for representing the **relations** among the atoms of the new predicate definitions

Example

APP with Convex Polyhedra

New predicate definitions:

P1whlP2ite(A,B,X,Y,A1',B1',X1',Y_1',A,B,X,Y,A2',B2',X2',Y2') ←
X1'≤X2'-1, **P1whl**(A,B,X,Y,A1',B1',X1',Y1'), **P2ite**(A,B,X,Y,A2',B2',X2',Y2')

P1whlP2whl(A,B,X,Y,A1',B1',X1',Y1',A,B,X,Y,A2',B2',X2',Y2') ← X1'≤X2'-1, A≤B-1,
X2=X1+A, **P1whl**(A,B,X1,Y,A1',B1',X1',Y1'), **P2whl**(A,B,X2,Y,A2',B2',X2',Y2')

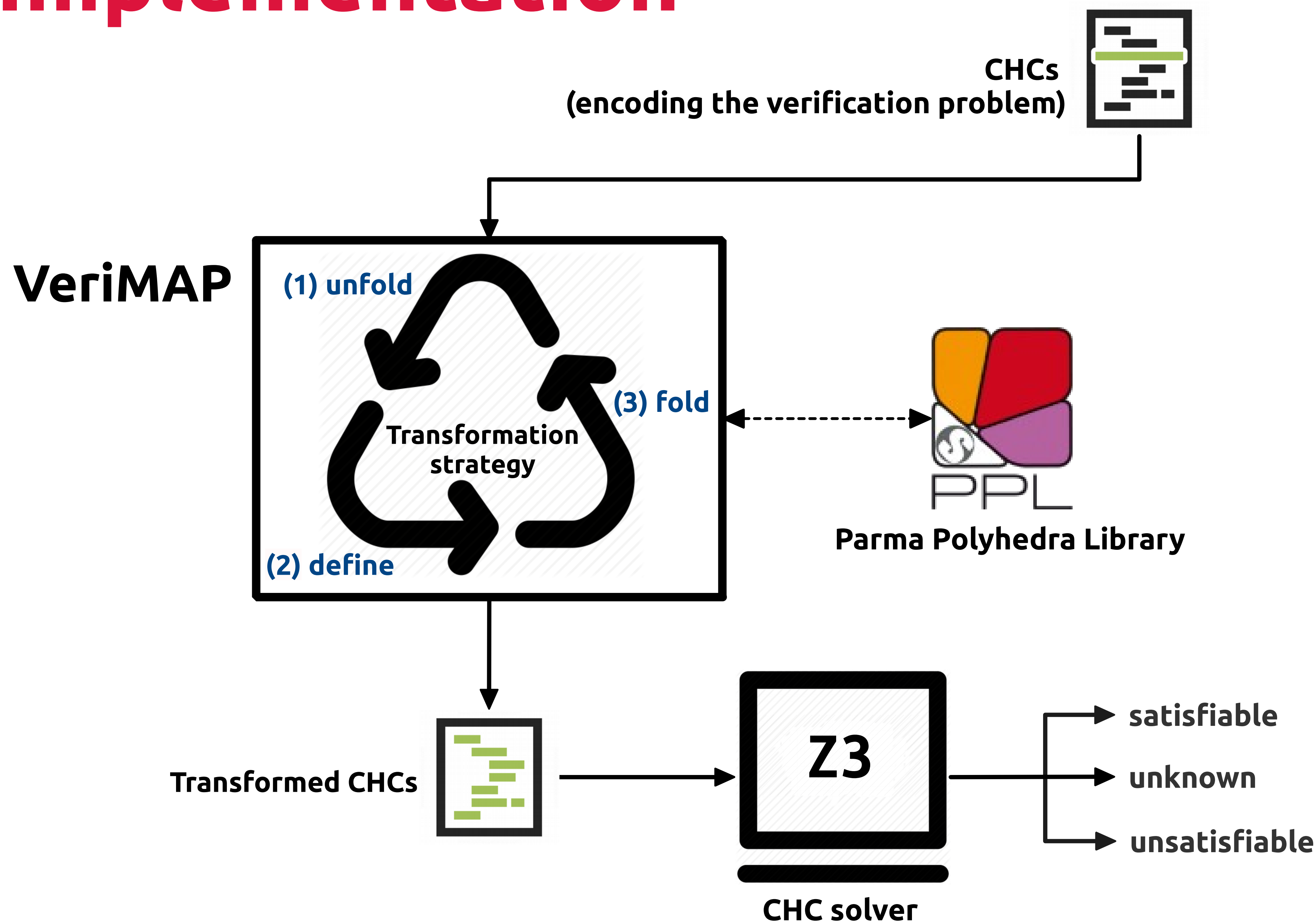
Final set of CHCs:

false ← A1=A2, B1=B2, X1=X2, Y1=Y2, X1'+1<=X2',
P1whlP2ite(A1,B1,X1,Y1,A1',B1',X1',Y1',A2,B2,X2,Y2,A2',B2',X2',Y2')

P1whlP2ite(A,B,C,D,E,F,G,H,A,B,C,D,I,J,K,L) ←
G≤K−1, A≤B−1, M=A+C,
P1whlP2ite(A,B,C,D,E,F,G,H,A,B,M,D,I,J,K,L)

P1whlP2whl(A,B,C,D,E,F,G,H,A,B,K,D,M,N,O,P) ←
G≤O-1, A≤B-2, K=A+C, R=A+1, T=A+C, S=D+T, X=A+1, W=K+X, Y=D+K,
P1whlP2whl(R,B,T,S,E,F,G,H,X,B,W,Y,M,N,O,P)

Implementation



Benchmark suite

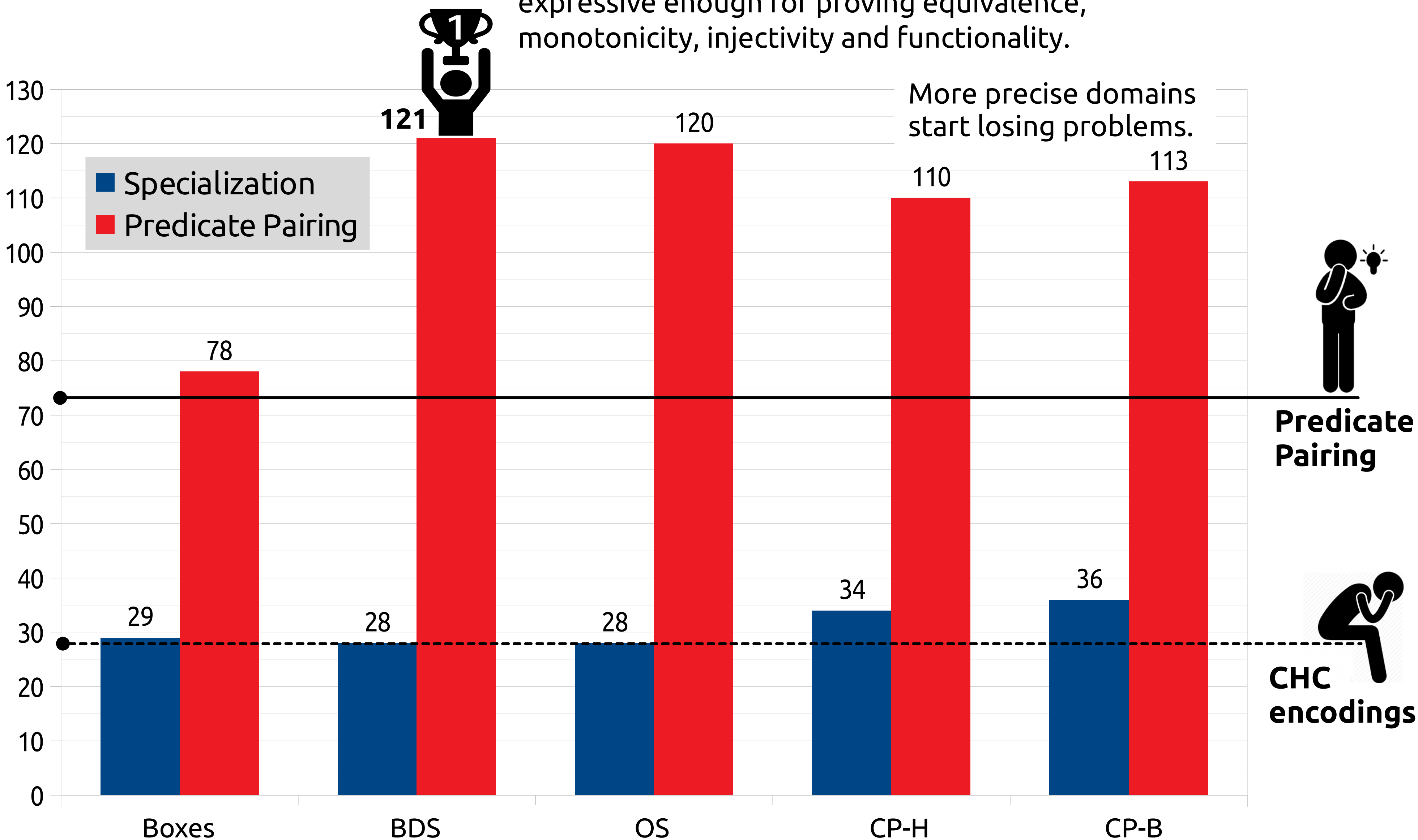
136	**Verification problems**
1655	**CHCs**

Relational properties

Equivalence	p1(X,X'), p2(Y,Y'), $X = Y \rightarrow X' = Y'$
Monotonicity	p(X,X'), p(Y,Y'), $X \leq Y \rightarrow X' \leq Y'$
Injectivity	p(X,X'), p(Y,Y'), $X' = Y' \rightarrow X = Y$
Functionality	p(X,f(X),X'), p(Y,f(Y),Y'), $X = Y \rightarrow X' = Y'$

Results

BDS is the best, followed by OS
expressive enough for proving equivalence, monotonicity, injectivity and functionality.



Specialization does not increase the number of problems solved and does not scale (polyvariant specialization causes a blow-up of the number of clauses)

Conclusions

A method for **combining**

- **transformation**
- **abstraction**

techniques, for proving **relational properties**

Improves **effectiveness** of state-of-the art **CHC solvers**

TODO: a finer control of the definition introduction to keep the size of transformed programs smaller